An Illustrative Approach to Use SQL Functions: A Review

Kamalinder Kaur
Assistant Professor
Chandigarh Engineering College
Punjab, India

***Abstract*—This paper describes the function used in databases for performing the calculations, modifies the data items and does manipulations on data. It defines the functions, their syntax's and errors occurred during the process. Functions are applied on oracle having SQL and are illustrated through query processing.**

Keywords- SQL; NVL; NULLIF; CASE; NVL2; DECODE.

I. INTRODUCTION

A SQL functions are brought up into oracle databases and are obtainable for its utilization in SQL queries.

II. BENEFITS OF SQL FUNCTIONS

The feature of SQL is its SQL Functions. These functions perform below tasks:

1) Executing calculations on data
2) Modification of individual data elements
3) Manipulate the results for collection of rows
4) Changing date and numbers to display
5) Conversion of data types of column

III. TYPES OF SQL FUNCTIONS

Single Row Function: These functions are applied on individual rows and then gives output on single row basis. The kinds of single-row functions are:

- On Character: Accepts character input and gives back both character and number digits.
- On Number: Accepts character input and gives back both character and numerical values.
- On Date: It works on values of the DATE data type. Almost all date functions outputs a value of DATE data type butthe MONTHS_BETWEEN gives a number.
- Conversion :altersvalue from one data type to another
- COALESCE, NVL, NULLIF, CASE, NVL2, DECODE are common functions.

Character Functions

Character cases handling functions: (Lower, Initcapand Upper)

TABLE 1: Character Functions

Function	Result
LOWER('Hello Word')	hello word
UPPER('Hello Word')	HELLO WORD
INITCAP('Hello Word')	Hello Word

IV. EXECUTION OF QUERIES

selectename"Emp_Name",lower(ENAME)"Lower_case",upper(ename) "Upper_case", initcap(ename) "Initcap_case" from emp1 where empno in ('1','2','3','4').

```
SQL> select ename"Emp_Name", lower(ENAME) "Lower_case",
upper(ename) "Upper_case", initcap(ename) "Initcap_case"
from emp1  where empno in ('1','2','3','4')  2    3  ;

Emp_Name   Lower_case Upper_case Initcap_ca
---------- ---------- ---------- ----------
singh      singh      SINGH      Singh
preet      preet      PREET      Preet
Anderson   anderson   ANDERSON   Anderson
sachin     sachin     SACHIN     Sachin
```

Figure 1: Describing the Character functions

This can work along with where clause: Select ename"Emp_Name",upper(job) "Job", hiredate from emp1 where ename='Anderson';

```
SQL> Select ename "Emp_Name",upper(job) "Job"
  2  , hiredate from emp1 where ename='Anderson'

no rows selected
```

Figure 2: Where clause

Query with initcap,lower,upper function with where clause:

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

select ename "Emp_Name", upper(job) "Job", hiredate from emp1 where initcap (ename) ='Anderson';

```
SQL> select ename "Emp_Name",upper(job) "Job",
  2  hiredate from emp1 where initcap (ename) ='Anderson';

Emp_Name   Job       HIREDATE
---------- --------- ---------
anderson   ANALYST   21-JAN-15

SQL> select ename "Emp_Name",upper(job) "Job",
  2  hiredate from emp1 where lower(ename)='anderson';

Emp_Name   Job       HIREDATE
---------- --------- ---------
anderson   ANALYST   21-JAN-15

SQL> select ename "Emp_Name",upper(job) "Job",
  2  hiredate from emp1 where upper(ename)='ANDERSON';

Emp_Name   Job       HIREDATE
---------- --------- ---------
anderson   ANALYST   21-JAN-15
```

Figure 3: Where clause in Character functions

Character-Manipulation Functions:
CONCAT: Joins the strings.
SUBSTR: Extracts the measurement lengthwise of the sub string.
LENGTH: It shows how long the string is numerically.
INSTR: helps in finding the numbered position of anyalphabet which is used.
LPAD: It justified the charactersalong the right position.
RPAD: It justified the characters along the left position.
TRIM: Itcuts thefront and last charactersoutofa string.

TABLE 2: Character –Manipulation Functions

Function	**Result**
Concat('Data' ,'Structure')	DataStructure
Substr("Data Structure",1,4)	Data
Length("Data Structure")	13
Instr('DataStructure', 'S')	5
Lpad(salary,5,'*')	**240
Rpad(salary, 5, '*')	240**
Replace('BACK and BUE','B','BL')	BLACK and BLUE
Trim('D' from 'DataStructure')	ataStructure

Selectename "Name",job "Desg.", concat(ename, job) "Concate Fun." from emp1 where empno in ('1','2','3','4');

```
SQL> select ename "Name" ,job "Desg.", concat (ename, job)
  2  "Concate Fun." from emp1 where empno in ('1','2','3','4');

Name       Desg.     Concate Fun.
---------- --------- ------------------
SINGH      MANAGER   SINGHMANAGER
PREET      CLERK     PREETCLERK
SACHIN     MANAGER   SACHINMANAGER
ANDERSON   ANALYST   ANDERSONANALYST
```

Figure 4: Concatination Function

use || symbol for cancat:
select ename "Name" ,job "Desg.", ename || ' is ' || job "Concate Fun." from emp1 where empno in ('1','2','3','4')

```
SQL> select ename "Name"
  2  ,job "Desg.", ename || ' is ' || job "Concate Fun."
  3  from emp1 where empno in ('1','2','3','4');

Name       Desg.     Concate Fun.
---------- --------- ----------------------
SINGH      MANAGER   SINGH is MANAGER
PREET      CLERK     PREET is CLERK
SACHIN     MANAGER   SACHIN is MANAGER
ANDERSON   ANALYST   ANDERSON is ANALYST
```

Figure 5: *use || symbol for cancat*

Example of Length and instr
select initcap(ename) "Name",initcap(job) "Desg.", initcap(ename) || ' is ' || initcap(job) "Concate Fun.",sal "Sal.",length(sal) "Length_sal",instr(ename,'a')"contains'a'",instr(upper(ename),'A') "contains lower 'a'"from emp1 where empno in ('1','3','4','7521')

```
SQL> select initcap(ename) "Name",initcap(job) "Desg.", initcap(ename) || ' is ' ||
  2  initcap(job) "Concate Fun.",sal "Sal.",length(sal) "Length_sal",
  3  instr(ename,'a')"contains'a'",instr(upper(ename),'A') "contains lower 'a'"
  4  from emp1 where empno in ('1','3','4','7521');

Name       Desg.     Concate Fun.                Sal. Length_sal contains'a' contains lower 'a'
---------- --------- ---------------------- ---------- ---------- ---------- ----------------
Ward       Salesman  Ward is Salesman            1250          4          0                1
Singh      Manager   Singh is Manager            2000          4          0                0
Sachin     Manager   Sachin is Manager            800          3          2                2
Anderson   Analyst   Anderson is Analyst         2500          4          0                1
```

Figure 6: Example of Length and instr

Example of SUBSTR,LPAD,RPAD
Selectename,substr(ename,1,4),sal,lpad(sal,10,'#'),rpad(sal,10,'#')from emp1

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

```
SQL> Select ename,substr(ename,1,4),sal,lpad(sal,10,'#'),
  2  rpad(sal,10,'#')from emp1;

ENAME      SUBS        SAL LPAD(SAL,1 RPAD(SAL,1
---------- ---- ---------- ----------
FORD       FORD       3000 ######3000 3000######
MILLER     MILL       1300 ######1300 1300######
SINGH      SING       2000 ######2000 2000######
PREET      PREE       5000 ######5000 5000######
Sachin     Sach        800 #######800 800#######
ANDERSON   ANDE       2500 ######2500 2500######
```

Figure 7: Example of SUBSTR,LPAD,RPAD

Example of Substr and Replace
selectename,substr(ename,1,3),replace(ename, 'a','u') from emp1 where ename like '%a%';

```
SQL> select ename,substr(ename,1,3),
  2  replace(ename, 'a','u') from emp1 where ename like '%a%';

ENAME      SUB REPLACE(EN
---------- --- ----------
Garyson    Gar Guryson
Sachin     Sac Suchin
anderson   and underson
```

Figure 8: Example of *Substr and Replace*

SQL statement displays the data for those employees whose last names end with the letter n.
Selectename,substr(ename,1,4),length(ename),instr(ename,'n') from emp1 where SUBSTR(ename, -1, 1) = 'n';

```
SQL> Select ename,substr(ename,1,4),length(ename),
  2  instr(ename,'n') from emp1 where
  3  SUBSTR(ename, -1, 1) = 'n';

ENAME      SUBS LENGTH(ENAME) INSTR(ENAME,'N')
---------- ---- ------------- ----------------
Garyson    Gary             7                7
Sachin     Sach             6                6
anderson   ande             8                2

SQL>
```

Figure 9: last names end with the letter n.

Number Functions:

TABLE 3: Number Functions

Function	Purpose
ROUND(*column|expression,n*)	This roundsoffvalues, cols and numerals upto n decimal places, if n is not included then no decimal places, if n is unsignedthenthe numerals to left position of decimal points are rounded off.
TRUNC(*column|expression, n*)	It eliminates the values to n decimal places, if n is non considerableit gives zero value.
MOD(*m,n*)	Gives leftovers of m by n.

Select round(45.923,2),round(45.923,1),round(45.923,-1), round(44.923,-1) from dual;

```
SQL> Select round(45.923,2),round(45.923,1),
  2  round(45.923,-1),round(44.923,-1) from dual;

ROUND(45.923,2) ROUND(45.923,1) ROUND(45.923,-1) ROUND(44.923,-1)
--------------- --------------- ---------------- ----------------
          45.92            45.9               50               40
```

Figure 10: Round function.

Select trunc(45.923,2),trunc(45.923,1),trunc(45.923,-1),trunc(44.923,-1) from dual;

```
SQL> select trunc(45.923,2),trunc(45.923,1)
  2  ,trunc(45.923,-1),trunc(44.923,-1) from dual;

TRUNC(45.923,2) TRUNC(45.923,1) TRUNC(45.923,-1) TRUNC(44.923,-1)
--------------- --------------- ---------------- ----------------
          45.92            45.9               40               40
```

Figure 11: Trunc function

selectename "Emp_Name",sal "Sal." , MOD(sal, 1000) from emp1 where empno in ('1','2','3','4');

```
SQL> select ename "Emp_Name",sal "Sal." , MOD(sal, 1000)
  2  from emp1 where empno in ('1','2','3','4');

Emp_Name         Sal. MOD(SAL,1000)
---------- ---------- -------------
SINGH            2000             0
PREET            5006             6
Sachin          10005             5
anderson         8007             7
```

Figure 12: Mod function

Operating Dates:
The Oracle recordsdates in an interiorsyntax:
Century-year-month-day-hours-minutes- seconds.
The automatic date demonstratesyntax is DD-MON-YY
selectename,hiredate from emp1;

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

```
SQL> select ename,hiredate from emp1;

ENAME      HIREDATE
---------- ---------
FORD       03-DEC-08
MILLER     23-JAN-10
SINGH      10-DEC-14
PREET      15-JAN-15
Garyson    05-DEC-12
Sachin     26-JAN-16
anderson   21-JAN-15
```

Figure 13: Hire date function

HIREDATE results as DD-MON-YY. This data is stored internally as follows:

Cen	Yr	Mon	D	Hr	Min	SEC
20	12	12	17	17	10	43

Calculation on Dates

1) The resultant date value can be added or subtracted to or from a date.
2) The no. of days can be calculated between two by subtracting them.
3) Hours to a date can be calculated by dividing the number of hours by 24.

selectename "Name" ,hiredate "Hire_date",hiredate+7 "7days+hiredate" ,hiredate-7 "7days-hiredate"from emp1

```
SQL> select ename "Name" ,hiredate "Hire_date",
  2  hiredate+7 "7days+hiredate",
  3  hiredate-7 "7days-hiredate" from emp1;

Name       Hire_date 7days+hir 7days-hir
---------- --------- --------- ---------
FORD       03-DEC-08 10-DEC-08 26-NOV-08
MILLER     23-JAN-10 30-JAN-10 16-JAN-10
SINGH      10-DEC-14 17-DEC-14 03-DEC-14
PREET      15-JAN-15 22-JAN-15 08-JAN-15
Garyson    05-DEC-12 12-DEC-12 28-NOV-12
Sachin     26-JAN-16 02-FEB-16 19-JAN-16
anderson   21-JAN-15 28-JAN-15 14-JAN-15
```

Figure 14: Hire date function

selectename "Name" ,hiredate "Hire_date", round((sysdate-hiredate)/7,0)"No.ofweeks",round((sysdate-hiredate)/30,0)"No.of months",round((sysdate-hiredate)/365,0)"No.of years" from emp1.

```
SQL> select ename "Name" ,hiredate "Hire_date",
  2  round((sysdate-hiredate)/7,0)"No.ofweeks",
  3  round((sysdate-hiredate)/30,0)"No.of months",
  4  round((sysdate-hiredate)/365,0)"No .of years"
  5  from emp1;

Name       Hire_date No.ofweeks No.of months No .of years
---------- --------- ---------- ------------ ------------
FORD       03-DEC-08        376           88            7
MILLER     23-JAN-10        317           74            6
SINGH      10-DEC-14         62           15            1
PREET      15-JAN-15         57           13            1
Garyson    05-DEC-12        167           39            3
Sachin     26-JAN-16          3            1            0
anderson   21-JAN-15         56           13            1
```

Figure 13: Round on Hire date function

Features of Date:

1) Months_Between (date1, date2):
   It helps us in finding the no. of months between two dates.If date1 is afterwarddate2---output is positive; if date1 is earlier than date2, the output is negative. The non-integer portion of the output shows a segment of the month.

2) Add_Months (date, n): Add no. of months into the existing calendar date. It works only on integer values and also can be negative.

3) Next_Day(date, 'char'): Locates the next day date after the given date. It gives output in character.

4) Last_Day (date): Discovers the end date of the month while considering the given date.

5) ROUND (date [,'fmt']): Yieldsrounding of the date to specified syntax. If the syntax fmt is neglected, then date is rounded of the nearby date.

6) TRUNC (date [, 'fmt']): It yields the date after the time truncated from it. If the syntax fmt is neglected, then date is truncated to the nearby date.

Selectename,hiredate,relievingdate,round(Months_Between(relievingdate,hiredate),0)"Exp._Month",Add_Months (hiredate,,Next_Day(hiredate,'SUNDAY'),Last_Day(hiredate) from emp1;

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

```
SQL> Select ename, hiredate, relievingdate,
  2  round(MONTHS_BETWEEN(relievingdate,hiredate),0)
  3  "Exp._Month",ADD_MONTHS (hiredate, 6),
  4  NEXT_DAY (hiredate,'SUNDAY'),
  5  LAST_DAY(hiredate)  from emp1;

ENAME      HIREDATE  RELIEVING Exp._Month ADD_MONTH NEXT_DAY( LAST_DAY(
---------- --------- --------- ---------- --------- --------- ---------
FORD       03-DEC-08 12-FEB-15         74 03-JUN-09 07-DEC-08 31-DEC-08
MILLER     23-JAN-10 15-FEB-16         73 23-JUL-10 24-JAN-10 31-JAN-10
SINGH      10-DEC-14 31-DEC-15         13 10-JUN-15 14-DEC-14 31-DEC-14
PREET      15-JAN-15 31-JAN-16         13 15-JUL-15 18-JAN-15 31-JAN-15
Garyson    05-DEC-12 05-JUL-13          7 05-JUN-13 09-DEC-12 31-DEC-12
Sachin     26-JAN-15 25-JAN-16         12 26-JUL-15 01-FEB-15 31-JAN-15
anderson   21-JAN-12 21-JAN-16         48 21-JUL-12 22-JAN-12 31-JAN-12
```

Figure 14: date function

In Where Clause
Selectename,hiredate,relievingdate,round(Months_Between (relievingdate,hiredate),0) "Exp._Month" from emp1
WhereMonths_Between (relievingdate,hiredate) >=12

```
SQL> Select ename,hiredate,relievingdate,
  2  round(MONTHS_BETWEEN (relievingdate,hiredate),0)
  3  "Exp._Month" from emp1
  4  where MONTHS_BETWEEN (relievingdate,hiredate) >=12;

ENAME      HIREDATE  RELIEVING Exp._Month
---------- --------- --------- ----------
FORD       03-DEC-08 12-FEB-15         74
MILLER     23-JAN-10 15-FEB-16         73
SINGH      10-DEC-14 31-DEC-15         13
PREET      15-JAN-15 31-JAN-16         13
anderson   21-JAN-12 21-JAN-16         48
```

Figure 15: Hire date function with where clause

Round and Truncate Function with Dates
Select ename, hiredate, ROUND(hiredate,'MONTH'), TRUNC(hiredate,'MONTH'), ROUND(hiredate,'YEAR'), TRUNC(hiredate,'YEAR'), ROUND(hiredate,'DAY'), TRUNC(hiredate,'DAY') FROM EMP1;

```
SQL> Select ename, hiredate, ROUND(hiredate,'MONTH'),
  2  TRUNC(hiredate,'MONTH'), ROUND(hiredate,'YEAR'),
  3  TRUNC(hiredate,'YEAR'), ROUND(hiredate,'DAY'),
  4  TRUNC(hiredate,'DAY') FROM EMP1;

ENAME      HIREDATE  ROUND(HIR TRUNC(HIR ROUND(HIR TRUNC(HIR ROUND(HIR TRUNC(HIR
---------- --------- --------- --------- --------- --------- --------- ---------
FORD       03-DEC-08 01-DEC-08 01-DEC-08 01-JAN-09 01-JAN-08 30-NOV-08 30-NOV-08
MILLER     23-JAN-10 01-FEB-10 01-JAN-10 01-JAN-10 01-JAN-10 24-JAN-10 17-JAN-10
SINGH      10-DEC-14 01-DEC-14 01-DEC-14 01-JAN-15 01-JAN-14 07-DEC-14 07-DEC-14
PREET      15-JAN-15 01-JAN-15 01-JAN-15 01-JAN-15 01-JAN-15 18-JAN-15 11-JAN-15
Garyson    05-DEC-12 01-DEC-12 01-DEC-12 01-JAN-13 01-JAN-12 02-DEC-12 02-DEC-12
Sachin     26-JAN-15 01-FEB-15 01-JAN-15 01-JAN-15 01-JAN-15 25-JAN-15 25-JAN-15
anderson   21-JAN-12 01-FEB-12 01-JAN-12 01-JAN-12 01-JAN-12 22-JAN-12 15-JAN-12
```

Figure 16:Round and Truncate on date

Conversion Functions

If Oracle server needs to convert one data type to the other then it can repeatedly .Converts the data to expected data type. The expected data type by the Oracle server conversion can occur wholly and clearly by the user. For this purpose some functions are required to forcefully convert the data casting to another known as conversion functions. The function names follow the conventional input data type TO output data type.

1) Conversion Type: Implicit Data Type

CHAR, VARCHAR2 can be wholly changed to NUMBER or DATE. NUMBER type value can be routinely converted to character data by Oracle server. It occurs only when the character signifies a valid number or date type value correspondingly.

For example : the select queries outputs same because Oracle inside allows 1000 and '1000' as same.
Query-1
SELECT ENAME,JOB,SAL
FROM EMP1
WHERE SAL >15000;
Query-2
SELECT ENAME,JOB,SAL
FROM EMP1
WHERE SAL > '15000';

2) Conversion: Explicit Data Type

These functions are for single row which are skillful of converting column value, literal or an expression.
TO_DATE
TO_NUMBER
TO_CHAR

3) Function: TO_CHAR

It is required to cast a numeric input value to character type using a fixed model.

Format:
TO_CHAR(num1,[format],[nls_parameter])
Think about the below SELECT query. The query syntax the HIRE_DATE and SALARY columns of EMPLOYEES table using TO_CHAR ().

SELECT ENAME,TO_CHAR (hiredate, 'MONTH DD, YYYY') HIREDATE,TO_CHAR (sal, '$99999.99') Salary FROM emp1

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

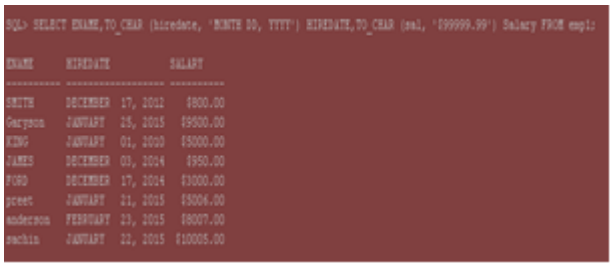

Figure 17: To-char function

TABLE 4: To-char function

Syntax Model	Explanation
,(comma)	This allots the position to a comma. Many no. of commas can be particularlyin a number syntax model. Boundaries: a number syntax model cannot begin by comma element and it cannot comeat the right arrangementof a decimal character or period.
.(period)	It gives the definite position. Boundaries:Itindicates only one function in a number layout model
$	Yieldsassessmentwith dollar sign.
0	It begins with zeros and proceedswith zeros at end.
	positive-gives value with the describednumber of digits with space in frontand negative with a minus sign in front.

5) Function:TO_NUMBER

Itconverts a numeric datatype from a character datatype.

Syntax:
TO_NUMBER(string1,[format],[nls_parameter])
list of layout models which can be used to typecast character values as number using TO_NUMBER.

Layout Model	Explanation
CC	Denotes Century
SCC	It gives Century Before Christstarted with -
YYYY	It displays year havingfour numbers
SYYY	It gives year before Christ with-prefixed with -
IYYY	Gives ISO Year havingfour numbers
YY	It is Year having 2 digits
YEAR	Gives Year in alphabets
SYEAR	Yields Year in alphabets, BC prefixed with -
MONTH	Gives Month in alphabets(i.e. January)
MON	Results JAN, FEB
WW	Gives Week number (i.e. 1)
W	Gives Week digit of the month (i.e. 5)
IW	Gives Week digit of the year in ISO standard.
DDD	Results Day of years in numbers (i.e. 365)
DD	Results month day in values (i.e. 28)
D	Gives week day in numbers(i.e. 7)
DAY	Gives Day of the week in alphabets (i.e. Monday)
FMDAY	Gives Day of the week in characters (i.e. Monday)
DY	Results Day of the week in short character description (i.e. SUN)
J	It Yields Julian Day
HH,H12	Gives Hour number of the day (1-12)
AM, PM	Gives AM or PM
MI, SS	Denotes Number of minutes and seconds (i.e. 59) ,
SSSSS	Gives seconds number of day.
DL	Results Long date format. Depends on NLS-settings. Use only with timestamp.
EE	Gives the full period name
FF	Gives the fractional seconds. Use with timestamp.
FF1..FF9	Gives the fractional seconds. Use with timestamp.
FM	It Fill Mode.
FX	It Format Exact: requires proper pattern matching between date and layout model.
RM	Returns The Roman cipherfor month (I .. XII)
RR	Returns The last 2 digits of the year.
RRRR	Returns The last 2 digits of the year when used for output. Accepts fout-digit years when used for input.
TH	It Converts a integer to it's ordinal layout. For example 1 becomes 1st.
TS	Gives Short time format. Depends on NLS-settings. Use only with timestamp.
TZD	It is reduced time zone name. ie PST.
TZR	Denotes Time zone region
X	It Denotes Local radix character. It is a period (.) in America

The SELECT queries written beneathallow numbers as alphabet intake.

SELECT TO_NUMBER('12,100.73', '999999.99') FROM DUAL;

2) Function:TO_DATE

This acceptsalphabet values as intake and outputs theplanned date. The TO_DATE function permits users to use a date in any layout, and then it reverts the input into the default layout used by Oracle 11g.
Syntax:
TO_DATE(string1, [format_mask], [nls_language])

TABLE 5: To-date function

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

Layout Model	Explanation
YEAR	It spelled outYear
YYYY	Gives 4-digit year
IYY,IY,I	Gives Last 3, 2, or 1 digit(s) of ISO year.
IYYY	Four -digit year based on the ISO standard
Q	It gives Quarter of year (1, 2, 3, 4; JAN-MAR = 1).
MM	Returns Month (01-12; JAN = 01).
MON	Gives name of month.
MONTH	Results Name of month, covering with blanksupto 9 characters.
RM	Gives Roman numerals for month starting from I-IX.
WW	Returns Week of year (1-53)
W	Gives Week of month (1-5)
IW	On the basis of ISO standard week of year is 1-52or1-53
D	Returns the week day.
DAY	Gives Name of day of week.
DD	Gives month day (1-31).
DDD	Gives year day (1-366).
DY	name of day is abbreviated
J	Returns Julian day;
HH12	Gives day hours (1-12).
HH24	Gives day hour(0-23).
MI,SS	Gives Minute (0-59).
FF	Returns seconds in fraction.
AM,PM	Gives indicator Prime Meridian
TZH,TZM,TZR	Results Time zone in hour, minute.

Example: a character string transforms into a date syntax.
SELECT TO_DATE('February 15, 1970, 11:00 A.M.', 'Month dd, YYYY, HH:MI A.M.', 'NLS_DATE_LANGUAGE = American')
FROM DUAL;

TO_DATE(15-FEB-70)

Common Functions
These are used to holdvoid values in database. The purpose of the common NULL controlling function is to swap the void values with a substitute value.

NVL
The NVL -deputies another value for a void value. NVL function can be used with all kinds of data types.

Syntax:
NVL(Arg1, replace_with)

This case includesboth the constraintswhich are mandatory.

The SELECT statement will display 'n/a' if an employee has not been assigned any job yet i.e. JOB_ID is NULL. Else, it would exhibit the actual JOB_ID value.

SELECT first_name, NVL(JOB_ID, 'n/a')
FROM employees;

NVL2
It is an improvement over the earlier NVL,Oracle presented a facility to standby data not only for NULL columns values but also for NOT NULL columns. NVL2 can be used an alternate for Null (Void) and also for non-null value.
Syntax:

NVL2(string1, value_if_NOT_null, value_if_null)

The SELECT statement under would display 'all' if the JOB_CODE for an employee is NULL. Finally, not null value of JOB CODE, it would rather display constant value 'Job done'.
SQL> SELECT NVL2(all, 'Job done', 'Bench')FROM employees;

NULLIF
The NULLIF is related to two arguments expr1 and expr2. If expr1 equals to expr2 then it gives NULL otherwise expr1. Dissimilar to it first parameter cannot be void.

Syntax:
NULLIF (expr1, expr2)
In this the first parameter can be nearer to NULL, but not as NULL. Both the constraints are compulsory for its execution.

The under query yields NULL until values, 16 are equal to each other.

Select NULLIF (16, 16) from dual;

Also, under query yields ‘ABC’ since both the strings are not equal.

SELECT NULLIF ('ABC', 'MOON')FROM DUAL;

COALESCE
It is basic form of NVL that gives the first non-void phrase in the parameter list. It requires minimum two parameters but there is no limit on its maximum limit.

Syntax:
COALESCE (stmt1, stmt2, ...stmt_n)
Considering the SELECT query. The first not null data served into address domain for the employee.

SELECT COALESCE (address1, address2, address3) Address FROM employees;

The functioning of coalesce function is like to IF..ELSIF..ENDIF construct.

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

```
IF address1 =!NULLthen
result := address1;
ELSIF address2 =!null THEN
result := address2;
ELSIF address3 =!null THEN
result := address3;
ELSE
result := null;
END IF;
```

Functions: Conditional

Two functions DECODE and CASE are used in SQL statement.

1. DECODE function:

The function is similar to conditional statement IF..THEN..ELSE .

Syntax:

DECODE (exp, srch, output [, search, result]... [, default])

DECODE checks in sequence. If equality occurs between statement and search parameter, and it yields the conforming result. If no matches occurs then null is defined. In case types mismatch then oracle within does likely inbuilt alteration to yield the results.Oracle says two null values can be same in case of decode function.

```
SELECT DECODE(NULL,NULL,'EQUAL','NOT EQUAL')
FROM DUAL;

DECODE
-----
EQUAL
```

If NULL expression is found, then Oracle returns output of first search as null. The No. of components are 255.

```
Select first_name, salary, DECODE (hire_date,
sysdate,'NEW JOINEE','EMPLOYEE') FROM employees;
```

CASE expression

Its mechanism logically similar to DECODE but varies in format and utilization.

Syntax:

```
CASE [ expression ]
When 1_condition … result_1
When 2_condition … result_2
……..
 When n_condition … result_n
   ELSE output
END
```

The determined number of parameters in a CASE expression are 255. Each WHEN ... THEN pair calculates as two arguments. To evade exceeding the limit, nested CASE expressions can be used so that the output_exp itself is a CASE expression.

```
Select first_name, CASE          when salary < 100 THEN
'GRADE 1'
                    when salary > 100 AND salary <
4000 then 'GRADE 2'
                    ELSE 'GRADE 3'
                END CASE
From employees;
ENAM          CASE
----  -------
Admin  GRADE 2
Jass  GRADE 3
Kumar  GRADE 1
```

V. CONCLUSION

The Query processing of SQL functions comprises of conversion functions has done in this paper .This showed the data manipulation ,formatting, general functions, conditional functioning and its transformation from inbuilt to forceful conversion. In future the work can be done on multiple row functions also.

REFERENCES

[1] Groff and Weinberg, "SQL-The Complete Reference

[2] Ivan Bayross, "SQL, PL/SQL-The Programming Language of Oracle.

[3] Koch and KelvinLoney,"Oracle 9i- Complete Reference".S

[4] Korth, "Database System Concepts".

[5] https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm.

[6] https://msdn.microsoft.com/enus/library/gg415714.aspx.

[7] http://www.oracle.com/technetwork/database/bi-datawarehousing/wp-sqlnaturallanguageanalysis-2431343.pdf.

[8] https://ocw.mit.edu/courses/urban-studies-and-planning/11-521-spatial-database-management-and-advanced-geographic-information-systems-spring-2003/lecture-notes/lect4.pdf.

[9] Jean Habimana," Query Optimization Techniques - Tips For Writing Efficient And Faster SQL Queries " International Journal of Scientific & Technology Research volume 4, issue 10, october 2015 issn 2277-8616 22 ijstr©2015 .

[10] Swati Jain and Paras NathBarwa," Performance Analysis of Optimization Techniques for SQL Multi Query Expressions Over Text Databases in RDBMS ",International Journal of Information & Computation Technology. ISSN 0974-2239 Volume 4, Number 8 (2014), pp. 841-852.

[11] MJ Egenhofer,"Spatial SQL: A query and presentation language" IEEE Transactions on knowledge and data engineering 6 (1), 86-95.

[12] Max J Egenhofer,"Query processing in spatial-query-by-sketch" in Journal of Visual Languages &Computing ,in volume 8and issue 4.

[13] J.Herring,R.Larsen and J.Shivakumar, "Extensions to the SQL Language to support Spatial analysis in a Topological Database" ,GIS/LIS 88,San Antonio,TX,1988,pp.741-750

[14] M.Egenhofer, "why not SQL!",International Journal of Geographiccal Information Systems ,vo.6,no.2,pp.71-85,1992.

ISBN: 978-0-9957075-4-2 *DOI: 10.24032/ijeacs/0203/05*

[15] M.Jones, "An introduction to Gofer", Department of Computer Science,YalleUniversity,Technical report,1991.

AUTHOR PROFILE

Kamalinder Kaur is working currently as Assistant Professor in Chandigarh Engineering College, Punjab, India. She has five years of teaching experience, her research interest includes Networking with specialization in Mobile Ad-hoc Network (MANET).